Zadanie č. 3

Strojové učenie a neurónové siete

Bc. Miroslav Malíšek

# Analýza zadania

Ibbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbeddddddddddddddddddddddddddddddddddddddddddddddfvhrficvorocnoonkocnrkcnrkcnrkcnrkcnrkcnrcknr

# Časť zadania (Spoznajte dáta)

Obrázky, ktoré sa použijú v tomto zadaní, sú rozdelené do dvoch množín (priečinkov) – trénovacie a testovacie. V oboch množinách sa nachádzajú rovnaké triedy, resp. typy zvierat. Celkovo ich je 90. Ku každému zo zvierat máme celkovo k dispozícii 60 obrázkov, z toho 54 je trénovacích a 6 testovacích. Spolu je to 5400 obrázkov. Tieto informácie som v kóde uložil do dataframe-u, aby to bolo prehľadné.

Nasledujúca koláž zobrazuje reprezentanta každej triedy. Zvolil som ho ako prvý obrázok každej triedy trénovacej množiny.

Obrázok, na ktorom je koláž, zbierka, fotomontáž, snímka obrazovky

Automaticky generovaný popis

## Analýza podľa ImageNet

Ako model som si vybral EfficientNetB2, ktorá dosahuje úspešnosť 80.1%, resp. 94.9% v TOP-5. Zároveň obsahuje 9.2 miliónov parametrov, čo je rozumné číslo vzhľadom na dostupný hardvér (tento model použijeme aj v ďalších častiach zadania).

Predikcie získam zo všetkých dostupných obrázkov (trénovacie aj testovacie), takže vyhodnotenie prebieha na 5400 obrázkoch.

Nakoľko model nie je prispôsobený na naše výstupy, nemohol som priamo porovnávať predikované zaradenie obrázku do triedy s reálny zaradením. Často sa totiž stalo, že model to vyhodnotil správne, akurát názov zvieraťa bol iný, ako u nás. Prípadne predikoval už aj poddruh zvieraťa. Ak by sme to priamo porovnávali, prišli by sme o cenné informácie, ktoré vieme získať len ručným vyhodnotením.

Vyhodnotil som to teda tak, že pre každú našu triedu som zobrazil 3 najpočetnejšie predikcie modelu (resp. menej ak model predikoval pre našu triedu všetko do jednej, resp. dvoch tried). Výsledky som uložil do dataframe-u, ktorý vyzerá nasledovne:

Obrázok, na ktorom je text, snímka obrazovky, jedálny lístok, softvér

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, jedálny lístok

Automaticky generovaný popisObrázok, na ktorom je text, jedálny lístok, snímka obrazovky

Automaticky generovaný popis

Obrázok, na ktorom je text, snímka obrazovky, jedálny lístok

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, jedálny lístok

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, jedálny lístok, softvér

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, jedálny lístok, softvér

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, jedálny lístok, softvér

Automaticky generovaný popis

Stĺpec vľavo reprezentuje naše triedy zvierat (pravdivé), stredný predpovedané modelom a stĺpec vpravo zobrazuje, koľkokrát model predpovedal hodnotu v strednom stĺpci pre pravdivú triedu zvierat vľavo.

Pre každé zviera sme mohli mať maximálny počet predpovedí 60, takže najlepšie je, ak je číslo vpravo práve 60. To sa podarilo v jedinom prípade a to pre zviera *wombat*. Iné predikcie teda pre toto zviera už model nedal. Pre dve zvieratá predpovedal model 59x rovnakú triedu, konkrétne pre *orangutan* a*panda*. V prípade pandy však vidíme, že model predpovedal „giant panda“, takže tento prípad potvrdzuje to, že sme to nemohli presne porovnávať. Vyšlo by nám, že pre pandu sa model netrafil ani raz, čo by nebola pravda. V rozmedzí 50 až 58 model predpovedal rovnakú triedu v týchto zvieratách: *badger, bison, boar, chimpanzee, cockroach, coyote, eagle, flamingo, fly, goldfish, goose, gorilla, hare, hippopotamus, hornbill, hummingbird, hyena, jellyfish, koala, leopard, lion, lobster, otter, pelican, penguin, pig, porcupine, squirrel, starfish, tiger, zebra.* Spolu teda model predpovedal 50 alebo viackrát rovnako pri 34 zvieratách.

Pri hlbšej analýze by sme ešte zistili, že sú zvieratá, pri ktorých model síce nepredpovedal až tak veľakrát jednu triedu, avšak predpovedal už aj druhy jednotlivých zvierat, ako napríklad v prípade *bear, crab, elephant* alebo *shark*. Ak by sme brali druhy ako jedno zviera, tak v týchto prípadoch by sme sa tiež dostali nad 50 správnych predpovedí.

Na druhú stranu, pri niektorých zvieratách sa model výrazne mýlil. Uvediem len tie prípady, v ktorých sa v troch najčastejších predpovediach nevyskytuje ani raz správna predpoveď. Sú to *bat cow, donkey, goat, mosquito, mouse, octopus, okapi, oyster, pigeon, possum, raccoon, rat, rhinoceros, seahorse, squid, woodpecker.* Je pravdepodobné, že model žiadne obrázky podobné týmto zvieratám predtým nevidel a teda nebol ich schopný správne zaradiť.

Medzi najviac podobné triedy podľa predpovedí patria *antelope – deer* (predpovede gazelle), *starfish – octoupus* (predpovede starfish), *badger – possum* (predpovede badger), *dolphin – whale* (predpovede gray whale a killer whale), *cow – ox* (predpovede ox). V niektorých prípadoch sú podobnosti zjavné (*antelope – deer, badger – possum*). V iných prípadoch je to ale najmä preto, lebo jedna z tých tried nie je na modeli natrénovaná a nemá svoju vlastnú triedu. Model ju teda priradil do inej triedy na základe nejakých znakov (*starfish – octopus* sú vodné zvieratá, *cow* nemá vlastnú triedu a najviac sa podobá z modelu práve na *ox*).

# Časť zadania

V tejto časti vytvárame a trénujeme konvolučnú neurónovú sieť na klasifikáciu zvierat.

Na vytvorenie trénovacej, validačnej aj testovacej množiny použijeme funkciu image\_dataset\_from\_directory() z knižnice Keras. Validačné dáta vytvoríme z trénovacej množiny v pomere 1:9. Veľkosti obrázkov nastavíme na 180x180 pixelov a batch size na 32. Trénovacie dáta zamiešame. Vystupné triedy budú vo forme Categorical, teda budú mať tvar ako po One – hot encodingu. Ako samotné obrázky použijeme tie dodané so zadaním, pričom vo funkcii sa využíva to, že sú uložené do priečinkov.

Nasledujúci kód teda slúži na vytvorenie generátoru dát:

Obrázok, na ktorom je text, snímka obrazovky, jedálny lístok

Automaticky generovaný popis

Následne si vytvoríme konvolučnú neurónovú sieť. Bude obsahovať 2 konvolučné vrstvy, čo znamená 2 vrstvy Conv2D + MaxPooling2D.

Ešte pred prvou konvolučnou vrstvou normalizujeme vstupy do siete použitím preprocesnej vrstvy Rescaling. Samotné množiny teda upravovať normalizáciou nebudeme.

V prvej vrstve som zvolil počet filtrov 16 a kernelovú dimenziu 3x3. V druhej vrstve zase 32 filtrov, kernelová dimenzia ostala rovnaká. Aktivačná funkcia bude v oboch vrstvách relu. Maximálnu veľkosť poolingu som v oboch vrstvách zvolil 3x3.

Nasleduje vrstva Flatten, ktorá predchádza plne prepojeným vrstvám.

Plne prepojené vrstvy sú tvorené vrstvami typu Dense, pričom ich počet a aj počet neurónov v každej z nich už bude závisieť od konkrétnej realizácie siete. Budú tú uvedené rôzne konfigurácie. Avšak každá skrytá vrstva má aktivačnú funkciu relu a posledná, výstupná má aktivačnú funkciu softmax s počtom neurónov 90 (toľko máme rôznych druhov zvierat, resp. tried).

Ako kriteriálna funkcia je použitá categorical\_crossentropy. Je to kvôli tomu, že sa jedná o viactriednu klasifikáciu a výstupom teda bude one – hot vector s predikciami. Solver bude Adam a metriku budeme sledovať úspešnosť.

Do modelu zahrnieme tiež early\_stopping, ktorý bude monitorovať loss na validačnej množine a ak sa nebude znižovať po 3 za sebou idúcich epochách, tak sa trénovanie zastaví.

No natrénovaní modelu si zobrazíme grafy priebehov trénovania na trénovacej aj validačnej množine. Pre trénovaciu a testovaciu množinu si zobrazíme aj konfúzne matice. Vzhľadom na vysokú početnosť tried sa budeme zameriavať iba na farebnosť diagonály.

Konfigurácie:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Augmentácia | 1. Conv2D | 1. Conv2D | Dropout | Počet neurónov v skrytých vrstvách | Úspešnosť na trénovacej množine | Úspešnosť na testovacej množine |
| 1. | - | - | - | - | 256, 128 | 0.82 | 0.30 |
| 2. | - | L2(0.1) | L1(0.1) | - | 256, 128 | 0.082 | 0.053 |
| 3. | Horizontal flip, rotation (0.1) | L2(0.01) | - | - | 128 | 0.38 | 0.28 |
| 4. | Horizontal flip, rotation (0.1) | L2(0.1) | L2(0.001) | Dropout(0.2) po 1. Conv2D | 128 | 0.25 | 0.20 |

V prvom trénovaní som nepoužil žiadne regularizátory a počet neurónov som sa snažil nájsť taký, aby vzniklo veľké pretrénovanie. Vidíme, že rozdiel v úspešnosti je viac ako 0.5 v prospech trénovacej množiny, takže model sa výrazne pretrénoval.

V druhom som neurónové vrstvy nechal nezmenené a zaviedol som 2 regularizátory – v prvej konvolučnej vrstve L2 (0.1) a v druhej L1 (0.1). Podľa úspešnosti ale vidíme, že táto kombinácia bola príliš silná a nedovolila sa modelu učiť. Po poslednej epoche sa úspešnosť na trénovacích dátach dostala iba na 0.082 a na testovacích 0.053.

V treťom trénovaní som pridal tiež augmentáciu obrázkov ako prvú vrstvu konvolučnej siete. Konkrétne som aplikoval horizontálne otočenie a tiež rotáciu s faktorom 0.1. V prvej konvolučnej vrstve som zaviedol regularizátor L2(0.01). Okrem toho, počet skrytých vrstiev som znížil na jednu s počtom neurónov 128. Výsledkom bolo znížené pretrénovanie modelu. Na trénovacích dátach sa dosiahla úspešnosť 0.38 a na testovacích 0.28. Rozdiel je teda 0.1, čo sa blíži k prijateľnej hranici, ktorú by sme chceli dosiahnuť.

V poslednom štvrtom trénovaní som ponechal rovnakú augmentáciu, v prvej konvolučnej vrstve som použil L2(0.1) a v druhej L2(0.001). Taktiež som medzi prvú a druhú vrstvu vložil vypnutie neurónov Dropout v pomere 0.2. Plne prepojená vrstva ostala rovnaká ako v treťom trénovaní. Pri tejto konfigurácii sme dosiahli úspešnosť na trénovacej množine 0.25 a na testovacej 0.20. Úspešnosti síce neboli príliš vysoké, v predošlých sme dokázali získať lepšie výsledky, avšak tu sme výrazne minimalizovali pretrénovanie, čo bolo hlavným cieľom.

Nasledujúce obrázky sa viažu k najlepšiemu trénovaniu, ktoré je práve posledné, štvrté.

Obrázok, na ktorom je text, diagram, vývoj, rad

Automaticky generovaný popisObrázok, na ktorom je text, vývoj, rad, snímka obrazovky

Automaticky generovaný popis

Obrázok, na ktorom je text, rad, písmo, snímka obrazovky

Automaticky generovaný popis

Obrázok, na ktorom je text, snímka obrazovky, rad, diagram

Automaticky generovaný popis

Prvé dva obrázky zobrazujú grafy úspešnosti a chyby na trénovacej a validačnej množine v priebehu trénovania. Vidíme, že rozdiely nie sú veľké a krivky sú k sebe blízko. S ďalšími epochami by úspešnosť na oboch množinách pravdepodobne stále rástla, avšak pri loss vidíme, že na trénovacích začína postupne klesať rýchlejšie ako v prípade validačných. Tento rozdiel by sa pravdepodobne ďalej prehlboval. Avšak v rámci 15 epoch je to takmer ideálny priebeh.

Ďalšie dva obrázky ukazujú konfúzne matice pre trénovacie a testovacie dáta. Na oboch je diagonála zreteľne viditeľná, takže tento cieľ sme splnili. Na trénovacích je však predsa len trochu viditeľnejšia, keďže úspešnosť na týchto dátach je o 0.05 vyššia. Nie je to však príliš veľký rozdiel, takže aj toto potvrdzuje, že model nemá takmer žiadne pretrénovanie.

# Časť zadania

## Zhlukovanie

Najskôr si uložíme príznaky do dataframe-u. Vytvoríme nové generátory dát, jeden pre trénovacie dáta a jeden pre testovacie (kvôli rozdeleniu do prienčinkov) – na konci ich spojíme do jedného celku. Ani jednu množinu nemiešame, nie je to potrebné. Zároveň si z generátorov extrahujeme cesty k jednotlivým obrázkom.

Predtrénovanú sieť použijeme rovnakú ako v prvej časti zadania, teda EfficientNetB2, akurát s tým rozdielom, že bez vrchnej klasifikačnej vrstvy.

Pre obe množiny dát zavoláme metódu predict, ktorá nám pre model bez vrchnej vrstvy vráti pre každý obrázok maticu príznakov o rozmere (6, 6, 1408). Následne funkcia GlobalAveragePooling2D pomocou priemeru konvertuje danú maticu na vektor dĺžky 1408. Takýto vektor získame pre všetky obrázky z oboch množín a uložíme ich ako riadky do dataframe-u. Okrem toho posledné dva stĺpce budú obsahovať cestu k obrázku, pre ktorý sme daný vektor príznakov získali a tiež jeho triedu. Na záver už len spojíme takéto dva dataframy pre trénovacie, resp. testovacie dáta do jedného spoločného dataframu. Tento dataframe teda bude mať 1410 stĺpcov (počet príznakov + cesta + trieda) a počet riadkov 5400.

### Zhlukovací algoritmus

Vyberal som si medzi algoritmami K-means a DBSCAN. Pri oboch je potrebné určiť isté parametre, ktoré sa ale z našich dát určujú ťažko, keďže sú dosť abstraktné. Použil som teda všeobecné metódy, ktoré sa zvyknú používať na určenie parametrov.

Pri K-means je to parameter k, teda počet klastrov. Známa je Elbow metóda, v ktorej ide o to, že sa K-means vykoná pre všetky k z určitého intervalu a zistí sa kompaktnosť jednotlivých klustrov. Následne sa tieto hodnoty zobrazia do grafu a v ideálnom prípade by mal vyzerať ako „lakeť“ a hodnota v tomto lakti bude nové k. V našom prípade som však žiadny lakeť nevedel dostať. K som skúšal v intervale od 1 do 25. Skúšal som vstupné dáta bez úpravy, normalizovať, škálovať alebo redukovať pomocou PCA, avšak vo všetkých prípadoch vyšiel podobný graf ako tento:

Obrázok, na ktorom je text, vývoj, rad, snímka obrazovky

Automaticky generovaný popis

Nedá sa z neho presne určiť parameter k. Preto som musel skúsiť iný algoritmus, či pri ňom získam niečo relevantné.

DBSCAN je tzv. density – based algoritmus, ktorý požaduje 2 parametre. Prvým je minimálny počet vzoriek (min\_samples) potrebných pre vytvorenie klastra. Druhým je epsilon, ktorý určuje maximálnu vzdialenosť dvoch bodov, aby stále patrili do rovnakého klastra. Tu opäť existuje metóda, ktorá dokáže tieto parametre ukázať. Min\_samples nastavíme na dvojnásobok dimenzie datasetu, čo je v našom prípade 2x1408 = 2816. Následne sa vypočíta priemerná vzdialenosť medzi každým bodom a jeho k najbližšími susedmi (k = 2816). Tieto vzdialenosti sa potom vzostupne zobrazia do grafu. Vznikne krivka, z ktorej určíme epsilon tak, že určíme bod najväčšieho zahnutia a hodnota na y osi bude hľadaným epsilonom. Opäť som tento postup vyskúšal na viacerých úpravách datasetu a takéto sú výsledky:

Obrázok, na ktorom je text, rad, diagram, vývoj

Automaticky generovaný popisObrázok, na ktorom je text, rad, diagram, vývoj

Automaticky generovaný popis

Neupravený dataset Škálované vstupné dáta

Obrázok, na ktorom je rad, diagram, vývoj, text

Automaticky generovaný popisObrázok, na ktorom je text, rad, diagram, vývoj

Automaticky generovaný popis

Normalizácia vstupných dát Normalizácia + PCA (var 0.95)

Ako vidíme, grafy vyzerajú takmer identicky. Líšia sa len v hodnotách na osi y (vzdialenosť). Avšak, už len min\_samples tým, že vyšiel 2816, tak to napovedá, že to asi nie je správny postup. Pri tejto metóde sa počíta s tým, že dimenzionalita nebude vzhľadom na počet vzoriek taká veľká, ako v našom prípade. Nepomôže tomu ani redukcia PCA, nakoľko ak chceme zachovať dostatočnú varianciu, tak dimenzii bude stále príliš veľa. Túto metódu teda nemôžem použiť.

Tretia metóda sa viaže opäť ku K-means a volá sa Silhouette metóda. Táto metóda sa používa, ak nepomôže elbow metóda. Funguje na princípe tzv. Silhouette skóre, ktoré je vypočítané na základe intra klastrovej a inter klastrovej vzdialenosti. Knižnica Yellowbrick poskytuje implementáciu tejto metódy aj výstupným grafom. Nasledujúci obrázok zobrazuje práve takýto graf, ktorý bol vytvorený na vstupných dátach bez úpravy.

Obrázok, na ktorom je text, diagram, plán, rovnobežný

Automaticky generovaný popis

Nachádza sa tu 10 grafov pre hodnoty k od 2 do 11. Ideálne rozdelenie je také, ak všetky siluety dosiahnu vyššie ako priemerné skóre a taktiež ak je ich výška približne rovnaká. Tomu sa najviac približuje posledný graf, teda pre k = 11.

Nakoľko k = 11 je jediný odhad akéhokoľvek parametru zhlukovacích algoritmov, ďalej v zadaní použijem práve algoritmus K-means pre k = 11. Vstupné dáta budú bez úpravy.